Upper bound on DC wander

In newsletter v4_15 "When to Use AC Coupling" I included the following statement regarding the simulation of DC wander:

> *To estimate the degree of DC wander possible when passing a particular code through a certain high pass filter HPF(w), first set up a complimentary filter LPF(w), defined thus:*
>
> *LPF(w) = 1 - HPF(w)*
>
> *Then, pass the data code through the filter LPF(w) and look for the worst-case output. Whatever the magnitude of the output of LPF(w), that's the magnitude of the worst-case DC-wander error you will experience when passing the signal through HPF(w).*
>
> *The above simulation is made easier if you recognize that time constants associated with filter LPF(w) are usually long enough that the filter doesn't respond to individual bits. You can therefore simply input to the filter LPF(w) a sequence of DC-offset values (each value representing the DC-average value of a coded data word) without worrying about the exact pattern of bits encoded.*

That last statement about calculating the wander by only taking into account variations on a per-word basis is, as far as I can tell now, not true. The DC wander does indeed display small peaks within the body of each word that can be significant. Figure I shows the effect.

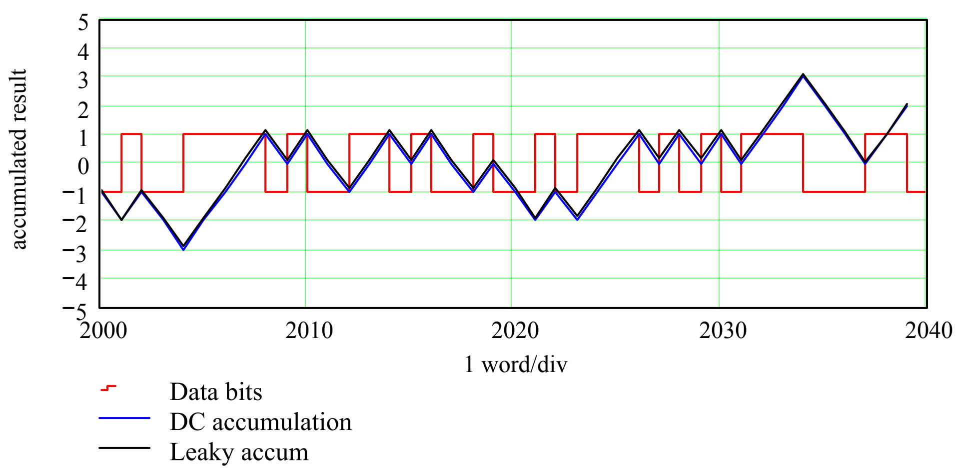

Figure I—Accumulation of DC wander by a perfect integrator (blue), showing the RD signal, and a leaky integrator (black) representing the output of a practical one-pole low-pass filter.

Ten-bit word boundaries upon which 8B10B guarantees a DC-balance range of +/1 are indicated with green vertical lines. As you can see, DC-wander ranges as far as +/-3 in this example. In a second example I've generated a "worst-case" packet that creates an excursion almost as big as 4.9, which I believe to be the absolute limit (Figure II).

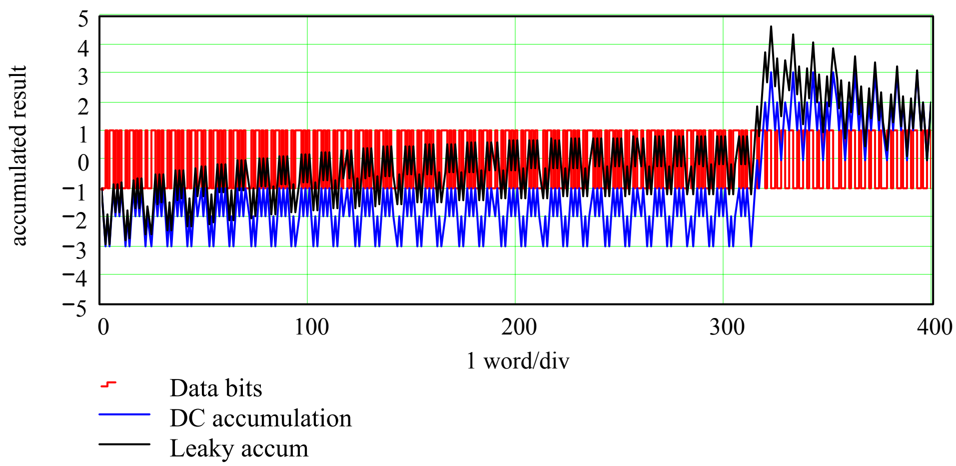

Figure II—Accumulation of DC wander for a worst-case test packet. Starting from RD- the packet comprises a coded sequence of words formed from twenty bytes of "84" (decimal notation for the data byte), followed by "252" and then "211".

In a practical system with AC-coupling time constant tau, coded data baud interval T, and signal amplitude $\pm A$, the worst-case DC-wander excursions will be bounded by $\pm 4.9A(T/\tau)$.

Here's how I arrived at the worst-case packet.

The system under analysis consists of the DC-wander estimator discussed in newsletter v4_15 "When to Use AC Coupling". Figure III illustrates the block diagram of this estimator.

The top of Figure III shows the input signal $x(t)$ coming into a low-pass filter, with output $z(t)$ representing the DC wander. What we seek are bounds on the amplitude of $z(t)$.

The low-pass-filter $\frac{1}{1+s\tau}$ factors into three portions: $\frac{1}{1+s\tau} = \frac{1}{sT}\frac{s\tau}{1+s\tau}\frac{T}{\tau}$.
This sequence of three block operations appears on the second line of the figure. The importance of this particular factoring is that the 8B10B code guarantees limits on the amplitude of the integrated signal $y(t)$. These limits are enforced by that code's DC-balance algorithm. In the terminology of this particular code the sum of all bits (assuming a one is valued at +1 and a zero at −1) is called the *running-disparity*, or RD.

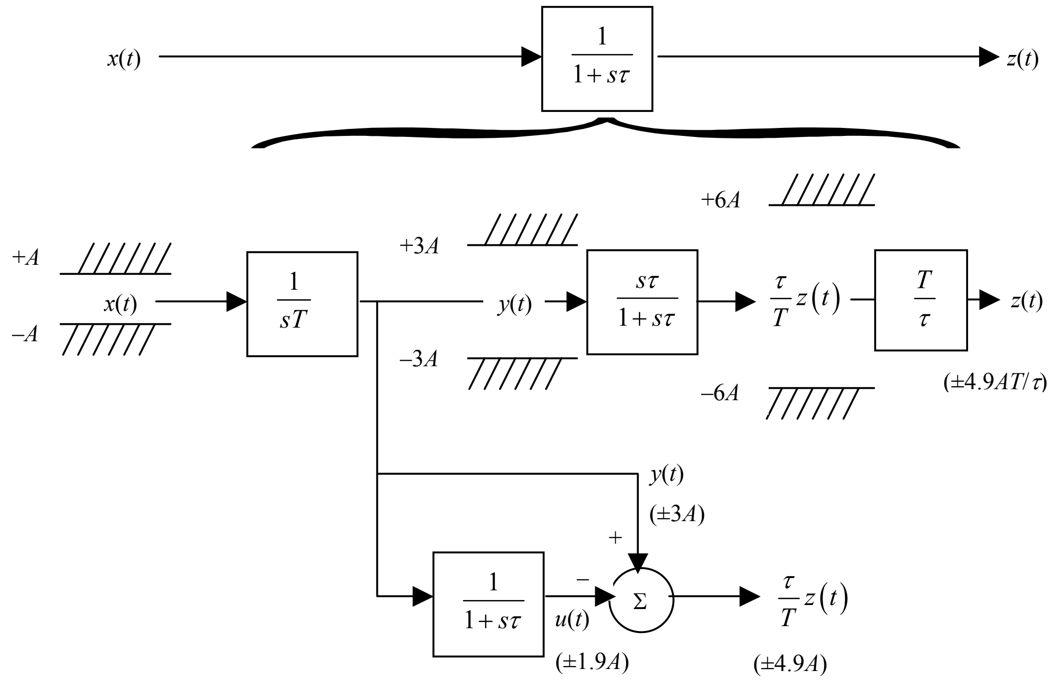

Figure III—Decomposition of DC-wander estimation filter leads to an expression relating worst-case RD excursions to worst-case DC wander.

The code by its construction guarantees that each 10-bit code word ends with RD +1 or −1 (never zero). These states are termed RD– and RD+.

The RD rules work like this:

1) Begin assuming RD–.
2) Most data bytes are coded into words having RD=0. Unfortunately, there aren't enough of these code words to fill an entire 256-byte space.
3) The remaining data bytes that can't have RD=0 are each assigned two data codings, one having RD=+2 and the other having RD=–2.
4) If the last code word ended in state RD–, then select the next code word from the table of words having either RD=0 or +2 .
5) If the last code word ended in state RD+, then select the next code word from the table of words having RD=0 or −2.
6) In summary, each code-word changes the RD by a value of +2, 0, or −2, keeping the result limited to the range [−1,+1].

Now, let's assume the code is in state RD+ and ask the question, what is the maximum temporary RD run-up that could occur during the transmission of a 10-bit code word? The answer to that question is 2, as exemplified by code-word number 243 (decimal), have coded value 1100100001. Sending the leftmost bit first, and starting with RD=1, the sequence of RD values produced by this code words are:

bit	(begin)	1	1	0	0	1	0	0	0	0	1
RD	1	2	3	2	1	0	1	0	−1	−2	−1

The peak value is 3. A similar analysis, starting with RD=–1 and using code word 252 (decimal) 0111100001 produces a worst-case trough of −3. You would get an even worse trough starting with RD=–1 if you could use the RD+ version of this same word, 1000011110, but that would be an illegal violation of the coding rules. If RD=–1 you have to use the RD– version of that code word. There are no worse cases (I checked them all, including the control words).

Summarizing our discussion of RD values, the worst-case RD peak and trough for all possible codings are +3 and −3, respectively. This coding arrangement principle establishes hard limits on the worst-case amplitude of $y(t)$. If the signal $x(t)$ has amplitude $\pm A$ then $y(t)$ is bounded to the range $\pm 3A$.

Next we must consider that the high-pass filter $\frac{s\tau}{1+s\tau}$ can no more than double the range of its input signal. This general statement applies to any high-pass filter $h(f)$ where the step response of the related low-pass filter $[1-h(f)]$ has a monotonic step response. Single-pole filters (and some other types of well-damped filters) fall into this category.

To see why that might be the case, I've re-drawn the high-pass filter operation as the difference between a low-pass filter $\frac{1}{1+s\tau}$ and a pass-through branch, according to the simple relation: $1-\frac{1}{1+s\tau}=\frac{s\tau}{1+s\tau}$.

The output range of the low-pass filter, if it is well damped (i.e., monotonic step response) cannot exceed the range of it's input. The range of $u(t)$ therefore must be bounded by $\pm 3A$, as is the range of $y(t)$. The difference between $u(t)$ and $y(t)$ therefore cannot exceed $\pm 6A$.

If the bound on $y(t)$ is $\pm 6A$, then after the last processing step (shown in the middle) the result $z(t)$ must be bounded by $\pm 6A\left(\frac{T}{\tau}\right)$

To derive the more restrictive bound of $\pm 4.9A\frac{T}{\tau}$ we have to look at some more code properties.

Specifically, beginning with state RD– ask yourself, "What code word, if repeated, would create the lowest possible average value, and thus the most extreme value of $u(t)$?". One answer to that question is code word 84 (decimal) 0010110101, producing a DC average RD value of -1.9. If you transmit a long string of 84's, then $u(t)$ eventually decays to near -1.9, and then if you quickly pop the RD up to +3 before $u(t)$ responds, you will see the difference illustrated in Figure III:

$$y(t)-u(t)=3.0-(-1.9)=4.9$$

Code word 171 (decimal) 1101001010 provides the same service starting with RD+, and produces a DC average value of +1.9. There are no worse cases (I checked them all, including the control words).

In conclusion, given bounds on RD of $\pm n$, DC wander will not exceed $\pm 2nA\frac{T}{\tau}$. In cases where more specific information is available about the worst-case DC average value of RD the bound may be reduced slightly (for 8B10B, to $\pm 4.9A\frac{T}{\tau}$).

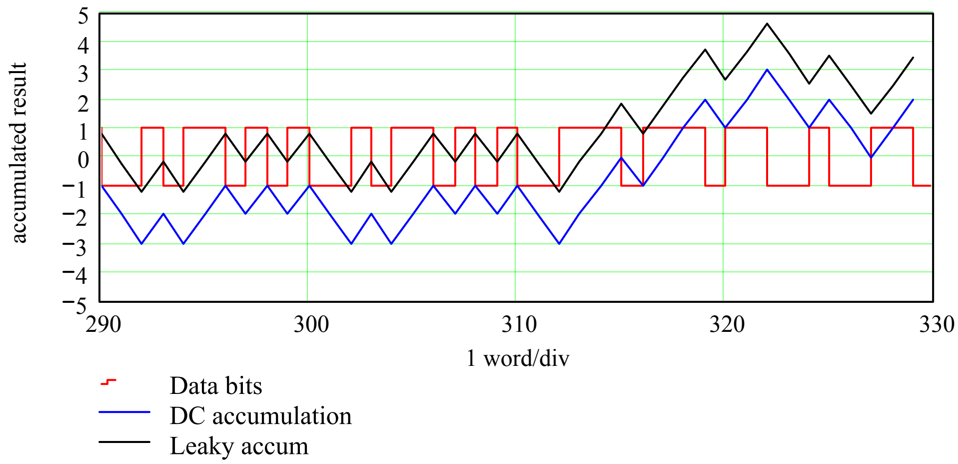

Figure IV—Detail view of Figure III. The data-byte sequence (prior to coding) is 84, 84, 252, 211.
